Yuzhda Yu., Samarai V.
(Igor Sikorsky Kyiv Polytechnic Institute, Kyiv)

THE DEVELOPMENT AND INVESTIGATION OF A GENETIC ALGORITHM FOR SOLVING THE DIOPHANTINE EQUATION

E-mail: juliaushda@gmail.com

In mathematics, a Diophantine equation is a polynomial equation, usually in two or more unknowns, such that only the integer solutions are sought or studied (an integer solution is a solution such that all the unknowns take integer values). A linear Diophantine equation equates the sum of two or more monomials, each of degree 1 in one of the variables, to a constant. An exponential Diophantine equation is one in which exponents on terms can be unknowns.

Diophantine problems have fewer equations than unknown variables and involve finding integers that work correctly for all equations. In more technical language, they define an algebraic curve, algebraic surface, or more general object, and ask about the lattice points on it.

The mathematical study of Diophantine problems that Diophantus initiated is now called Diophantine analysis.

Genetic Algorithms (GAs) are computer simulations to evolve a population of chromosomes that contain at least some very fit individuals. Fitness is specified by a fitness function that rates each individual in the population.

Setting up a GA simulation is fairly easy: we need to represent (or encode) the state of a system in a chromosome that is usually implemented as a set of bits. GA is basically a search operation: searching for a good solution to a problem where the solution is a very fit chromosome. The programming technique of using GA is useful for AI systems that must adapt to changing conditions because “re-programming” can be as simple as defining a new fitness function and re-running the simulation.

Investigation of a genetic algorithm for solving the Diophantine equation

Let us consider a diophantine (only integer solutions) equation: a+2b+3c+4d=30, where a, b, c, d are positive integers. Using a genetic algorithm, all that is needed is a little time to reach a solution (a, b, c, d). The architecture of GA systems allow for a solution to be reached quicker since "better" solutions have a better chance of surviving and procreating, as opposed to randomly throwing out solutions and seeing which ones work. First we will choose 5 random initial solution sets, with constraints $1 \leq a, b, c, d \leq 30$. (Note that we can choose smaller constraints for b,c,d, but for the sake of simplicity we shall use 30)

To calculate the fitness values, plug each solution set into the expression a+2b+3c+4d. Then, calculate the absolute value of the difference of each expression with 30, this will be our fitness value.

Since values that are lower are closer to the desired answer (30), these values are more desirable. In this case, higher fitness values are not desirable, while lower ones are. In order to create a system where chromosomes with more desirable fitness values are more likely to be chosen as parents, we must first calculate the percentages that each chromosome has of being picked. One solution is to take the sum of the multiplicative inverses of the fitness values (0,135266), and calculate the percentages from there (all simulations were created using a random number generator)

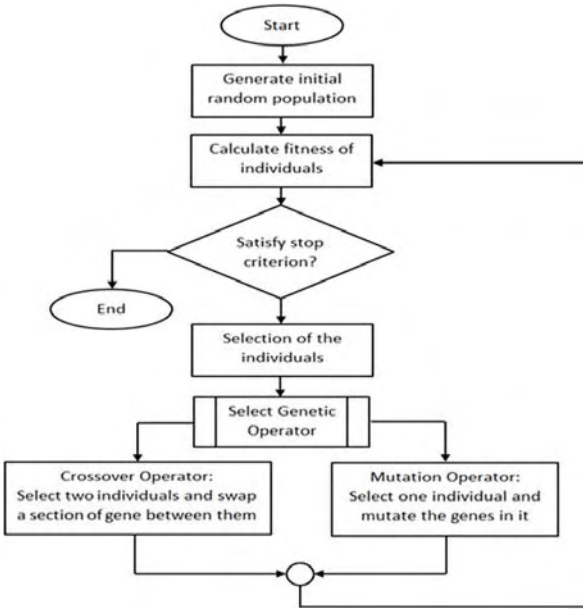

Fig. 1. Flow chart of the genetic programming approach

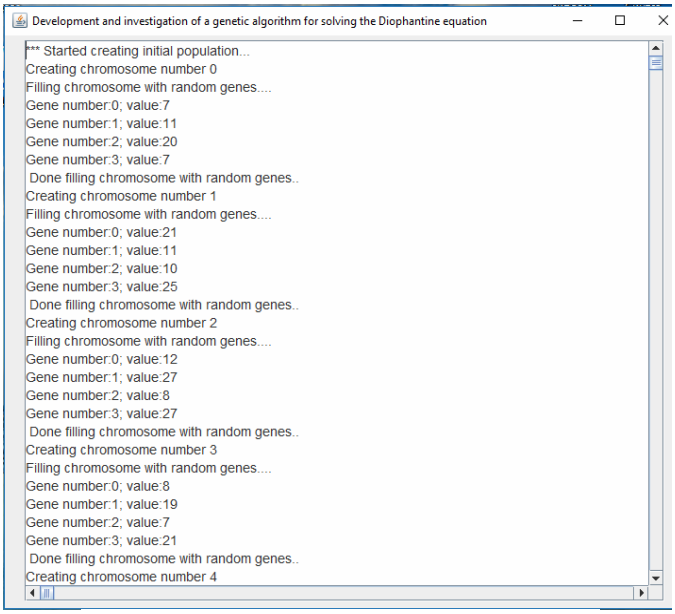

Fig. 2. Window of the developed program

The offspring of each of these parents contains the genetic information of both father and mother. How this can be determined is very arbitrary. However for this case, we could use something called a "cross-over". Let us say a mother has the solution set a_1,b_1,c_1,d_1, and a father has the solution set a_2,b_2,c_2,d_2, then there can be six possible cross overs (| = dividing line).

There are many other ways in which parents can trade genetic information to create an offspring, crossing over is just one way. Where the dividing line would be located is completely arbitrary, and so is whether or not the father or mother will contribute to the left or right of the dividing line.

Conclusion

The average fitness value for the offspring chromosomes were 38.8, while the average fitness value for the parent chromosomes were 59.4. Of course, the next generation (the offspring) are supposed to mutate, that is, for example we can change one of the values in the ordered quadruple of each chromosome to some random integer between 1 and 30. Progressing at this rate, one chromosome should eventually reach a fitness level of 0 eventually, that is when a solution is found. For systems where the population is larger (say 50, instead of 5), the fitness levels should more steadily and stabily approach the desired level (0).

Literary sources:

1. Practical Artificial Intelligence Programming With Java, Mark Watson, 2008, Version 3.0 United States License.

Zhbanova O., Saithareyev L.

(State Higher Educational Institution National University, Kryvyi Rih)

INVESTIGATION OF THE INFLUENCE OF ELECTRO-IMPULSE CURRENT ON MANGANIFEROUS LIQUID-ALLOY

E-mail: zhbanova.olena@gmail.com

The authors carried out a large volume of experimental studies on the influence of electro-impulse current in the process of crystallization of the casting during the study of steel grade 35GL.